NFORMATION AND COMPUTATION 86, 179–194 (1990)

On Measuring Nondeterminism in Regular Languages

JONATHAN GOLDSTINE

The Pennsylvania State University, University Park, Pennsylvania

C. M. R. KINTALA

AT&T Bell Laboratories, Murray Hill, New Jersey

AND

DETLEF WOTSCHKE*

Fachbereich Informatik, Johann Wolfgang Goethe-Universität, Frankfurt, West Germany

It is well known that allowing nondeterminism in a finite automaton can produce in the most extreme case an exponential savings in the number of states required to recognize a regular language. This paper studies situations intermediate between forbidding nondeterminism and allowing it. The amount of nondeterminism used by a finite automaton is quantified, so that the decrease in the size of the state space that occurs as the amount of nondeterminism that is permitted increases in increments can be studied. These intermediate situations are shown always to lie between two extremes:

(1) there are no savings as the amount of nondeterminism increases incrementally, so that savings occur only when the amount of nondeterminism becomes unlimited;

(2) each increment of nondeterminism results in additional savings, the number s of states decreasing approximately as $s^{1/i}$, until exponential savings have been achieved after about $i = \log s/\log\log s$ increments.

1. INTRODUCTION

Nondeterminism plays a profound role in the theory of automata. For certain types of automata, such as pushdown automata, it enhances their

* This author was supported in part by "Deutsche Forschungsgemeinschaft" under Grant Wo 334/2–1 and by "Stiftung Volkswagenwerk" under Grant II/62 325.

0890-5401/90 $3.00

power. For other automata, such as Turing machines and finite automata, nondeterminism does not enhance computational power but may increase efficiency. For Turing machines, the investigation of whether nondeterminism actually can result in savings of resources such as time or space leads to very difficult open problems, such as the P vs. NP and LBA problems. The present paper investigates the role of nondeterminism at the other end of the hierarchy of automata, that is, in finite automata. Time is not an issue since finite automata can always operate in real time. As for space, the only storage space available is in the finite-state control. Hence, the "resource" we measure is simply the number of states in the finite-state control. It is well known that nondeterminism can produce as much as an exponential savings in this resource (see Meyer and Fischer, 1971). In order to study more closely the trade-off of nondeterminism against other resources, it seems appropriate to treat nondeterminism as just one more resource by quantifying it. This was done for various automata by Kintala and Fischer (1980), and for finite automata by Kintala and Wotschke (1980). The present paper continues this study by introducing the concept of the "spectrum" of a regular language. This is a measure of the extent to which a regular language, viewed as a computational task for finite automata, can benefit from nondeterminism. The spectrum is thus an attempt to quantify the amount of nondeterminism inherent in a regular language rather than in a finite automaton.

Section 2 defines the spectrum of a regular language. Section 3 discusses the computability of spectra. Sections 4 and 5 establish nearly sharp upper and lower bounds on spectra. Section 6 lists some open questions.

2. The Spectrum of a Regular Language

Nondeterminism in a computation can be measured by counting the number of nondeterministic steps that occur in it. But, from an intuitive point of view, a step that selects one from among a large number of possible moves displays more nondeterminism than one that selects from fewer possibilities. Thus, we make the following definition.

Definition. A finite automaton over the alphabet Σ is a 5-tuple $A = (Q, \Sigma, \delta, q_0, F)$, with Q a finite set of states, $q_0 \in Q$, $F \subseteq Q$, and δ a function from $Q \times \Sigma$ to 2^Q. A *move* of A is a triple $\mu = (p, a, q)$ in $Q \times \Sigma \times Q$ with $q \in \delta(p, a)$. The *branching* $\beta_A(\mu)$ and *guessing* $\gamma_A(\mu)$ of the move μ are defined to be

$$\beta_A(\mu) = \#\delta(p, a) \qquad \text{and} \qquad \gamma_A(\mu) = \log_2(\#\delta(p, a)),$$

where $\#S$ denotes the cardinality of a set S. The language *accepted* by A is

$$L(A) = \{x_1 \cdots x_n \mid n \geqslant 1 \text{ and there exists a sequence of moves } (q_0, x_1, q_1),$$
$$(q_1, x_2, q_2), \ldots, (q_{n-1}, x_n, q_n) \text{ with } q_n \in F\} \cup \{\varepsilon \mid q_0 \in F\}.$$

The automaton A is an (incomplete) *deterministic* finite automaton if $\#\delta(p, a) \leqslant 1$ for all (p, a), or, equivalently, if $\beta_A(\mu) = 1$ $(\gamma_A(\mu) = 0)$ for every move μ. It is a *complete deterministic* finite automaton if $\#\delta(p, a) = 1$ for all (p, a). Branching and guessing are extended to computations $\mu_1\mu_2 \cdots \mu_t$, $t \geqslant 0$, by setting

$$\beta_A(\mu_1 \cdots \mu_t) = \beta_A(\mu_1) \times \cdots \times \beta_A(\mu_t)$$

and

$$\gamma_A(\mu_1 \cdots \mu_t) = \gamma_A(\mu_1) + \cdots + \gamma_A(\mu_t)^1.$$

For each word x in the language $L(A)$ defined by A, let

$$\beta_A(x) = \min \beta_A(\mu_1 \cdots \mu_t)$$

and

$$\gamma_A(x) = \min \gamma_A(\mu_1 \cdots \mu_t),$$

where $\mu_1 \cdots \mu_t$ ranges over all accepting computations of A with input x. When A is clear from the context, we omit it from the notation.

Intuitively, $\gamma(\mu)$ is the number of bits of information needed to single out and record the move μ from among the other moves that the automaton could have selected at any point in a computation where this move occurs. For a deterministic automaton, $\gamma(\mu)$ will always be zero; for an arbitrary automaton, it will be zero except for those moves at which guessing (nondeterminism) actually occurs. For a computation $\mu_1 \cdots \mu_t$, $\gamma(\mu_1 \cdots \mu_t)$ is a measure, in bits of information, of the amount of guessing occurring during the computation. If the automaton makes only binary guesses, so that $\beta(\mu_i)$ is always 1 or 2, then $\gamma(\mu_1 \cdots \mu_t)$ simply counts the number of nondeterministic moves in the computation. For a word x accepted by the automaton A, $\gamma(x)$ is the amount of guessing that A requires to recognize x when A is credited with its best effort, i.e., when A uses an accepting

[1] A string $\mu_1 \cdots \mu_t$ of moves of A is a *computation* if $\mu_i = (q_{i-1}, x_i, q_i)$, $1 \leqslant i \leqslant t$. Its *input* is $x_1 \cdots x_t$. It is an *accepting* computation if $q_t \in F$. When $t = 0$, $\mu_1 \cdots \mu_t$ is the empty computation; its input is the empty string ε; and it is an accepting computation if the start state q_0 is in F. By convention, $\beta_A(\mu_1 \cdots \mu_t) = 1$ and $\gamma_A(\mu_1 \cdots \mu_t) = 0$ for this computation.

computation for x that minimizes the amount of guessing. Note that crediting A with its best (rather than worst or average) performance is consistent with the way in which nondeterministic automata operate: they are credited with recognizing an input string if some computation leads to acceptance, even though other computations might fail to recognize the string. In addition, when $\beta(x)$ is defined as it is here by minimizing over all accepting computations of A on x, it reflects the amount of parallelism needed for a real-time simulation of A on x. This is so since all computations of A on x can be explored in parallel to a "depth" of $\beta(x)$ (that is, each branch can be explored up to the point at which the product of $\beta(\mu)$'s for all moves μ along the path exceeds $\beta(x)$), by $\beta(x)$ copies of A; and no branch need be explored deeper than this since acceptance is discovered by this point. (This is demonstrated formally in Lemma 5.2.) (An alternative approach, charging a nondeterministic automaton for its costliest successful effort on an input string, is taken by Savitch and Vermeir, 1981, in their study of nondeterminism in pushdown automata. While such a definition is simpler to work with technically, we do not believe that it is as well motivated.)

The behavior of $\gamma(x)$ can be more complex than may at first seem apparent. As x ranges over the language $L(A)$ accepted by A, $\gamma(x)$ is clearly bounded above by a linear function in the length $|x|$ of x. For each finite automaton A, it originally seemed reasonable to us to expect that either $\gamma(x)$ is bounded above by a constant or $\gamma(x)$ is a linear function of $|x|$ on an infinite subset of $L(A)$. Indeed, we succeeded in proving this to be true in the case where A has only a finite degree of ambiguity (Goldstine, Leung, and Wotschke, 1989). However, H. Leung (Goldstine, Leung and Wotschke, 1989) and I. Simon (1987) have shown independently that this is not true in general, and that, for each positive k, there is a finite automaton A for which $\gamma(x)$ is unbounded but $\gamma(x) = O(|x|^{1/k})$. In the present paper, instead of focusing on the total behavior of the function $\gamma(x)$ as a measure of the nondeterminism of A, we simply use the (possibly infinite) least upper bound on this function as a scalar measure of this nondeterminism. Thus, we make the following definition.

DEFINITION. If the language $L(A)$ accepted by the finite automaton A is empty, let $\beta_A = 1$ and $\gamma_A = 0$. Otherwise, let

$$\beta_A = \sup\{\beta_A(x) \mid x \in L(A)\}$$

and

$$\gamma_A = \sup\{\gamma_A(x) \mid x \in L(A)\}.$$

Thus, $\beta = \beta_A$ and $\gamma = \gamma_A$ measure the amount of nondeterminism that A requires to recognize $L(A)$. Note that $\beta = 2^\gamma$ whenever either is finite. The natural units in which to measure nondeterminism are bits, i.e., $\gamma = \log_2 \beta$, rather than the number of branches, β. But γ may fail to be an integer (unless we arbitrarily restrict attention to automata in which nondeterminism consists only of sequences of two-way choices). Hence, it is technically more convenient to use β as a measure of nondeterminism in what follows.

Since we intend to study the trade-off between the amount of nondeterminism and the size of a finite automaton, we make the following definition.

DEFINITION. Let $|A|$ be the number of states in the finite automaton A. The *spectrum* $\sigma(L)$ of a regular language L is the infinite sequence (with endpoint)

$$\sigma(L) = (\sigma_1(L), \sigma_2(L), ..., \sigma_i(L), ...; \sigma_\infty(L)),$$

where each $\sigma_i(L) = \min\{|A| : A \text{ is a finite automaton for } L \text{ with } \beta_A \leqslant i\}$. We omit L from the notation when no confusion results.

Note that $\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_\infty \geqslant 1$ for every spectrum. Since $\beta_A \leqslant \infty$ is always true, $\sigma_\infty(L)$ is simply the number of states in a minimal-state nondeterministic finite automaton for L. On the other hand, σ_1 is equal to the number of states in a minimal-state incomplete deterministic finite automaton for L. (This follows from the observation that, if $\beta_A = 1$, then the deterministic part of A, consisting of those moves μ for which $\beta(\mu) = 1$, accepts the same language as A.) If the standard subset construction is applied to a nondeterministic finite automaton with n states, yielding 2^n states, one for each subset of the original states, and if the dead state corresponding to the empty subset is excised, then an equivalent incomplete deterministic finite automaton with $2^n - 1$ states is produced. Hence, $\sigma_1 \leqslant 2^{\sigma_\infty} - 1$ for every spectrum.

Since each $\sigma_i \geqslant 1$, the smallest spectrum is the "trivial" spectrum $(1, 1, ...; 1)$. Obviously, the spectrum of a regular language L is trivial iff $L = \varnothing$ or $L = \Sigma^*$ for some finite alphabet Σ. (Here, Σ could be empty, i.e., $L = \{\varepsilon\}$ has a trivial spectrum.) In any nontrivial spectrum, $\sigma_j \geqslant 2$ for some j. Hence, $2 \leqslant \sigma_j \leqslant \sigma_1 \leqslant 2^{\sigma_\infty} - 1 \leqslant 2^{\sigma_i} - 1$ for all i, so $\sigma_i \geqslant 2$ for every entry in a nontrivial spectrum. Any language for which guessing does not help, i.e., for which $\sigma_1 = \sigma_\infty$, will have every entry in the spectrum the same. For example, it is not difficult to see that $L_n = (a^n)^*$ has spectrum $(n, n, ...; n)$ for $n \geqslant 1$. At the other extreme, for each $n \geqslant 1$, there are languages L_n for which nondeterminism can save an exponential number of states: $\sigma_\infty(L_n) = n$ and $\sigma_1(L_n) = 2^n - 1$ (Meyer and Fischer, 1971), so that

$$\sigma(L_n) = (2^n - 1, ?, ?, ?, ...; n).$$

In cases where nondeterminism does help, the middle entries of the spectrum can provide more detailed information about the trade-off between space and guessing. Sections 4 and 5 establish upper and lower bounds on the entries that can occur in the middle part of a spectrum.

3. Computability of the Spectrum

Although the spectrum of a regular language is an infinite sequence of integers, it is monotone and bounded, and so its entries assume just a finite number of values, each value assumed on an interval. Thus, the entire spectrum contains just a finite amount of information, and so we may ask not merely whether each entry in the spectrum of a regular language R is effectively computable from R but, more generally, whether the entire spectrum is. In this section, we show that the answer even to this more general question is yes.

3.1. Lemma. *The branching β_A of an automaton A is computable.*

Proof. If each move μ of A is given the weight 0 when $\gamma_A(\mu)=0$ and 1 when $\gamma_A(\mu)>0$, then γ_A is finite iff A is limited in distance in the sense of Hashiguchi (1982), who shows that there is a decision procedure for this property. (The proof is lengthy. This result was obtained independently by Leung, 1988, who gives an algebraic proof.) Hence, it is decidable whether or not β_A is finite.

For each positive integer i, let

$$L_i(A)=\{x \mid \text{there is an accepting computation } \pi \text{ of } A \text{ with input } x \text{ and with } \beta(\pi)\leqslant i\}.$$

To see that $L_i(A)$ is regular, suppose $A=(Q, \Sigma, \delta, q_0, F)$ and let Σ_T be the alphabet of triples corresponding to moves of A,

$$\Sigma_T=\{[p, a, q] \mid q\in\delta(p, a),\ p, q\in Q,\ a\in\Sigma\}$$

and let $R\subseteq\Sigma_T^*$ be the regular set of all accepting computations of A,

$$R=\{[p_0, a_1, p_1][p_1, a_2, p_2]\cdots[p_{t-1}, a_t, p_t]\in\Sigma_T^* \mid t\geqslant 1,\ p_0=q_0,\ p_t\in F\}\cup\{\varepsilon \mid q_0\in F\}.$$

Define homomorphisms $f\colon \Sigma_T^*\to\Sigma^*$ and $g\colon \Sigma_T^*\to\{c, d\}^*$ by

$$f([p, a, q])=a$$

$$g([p, a, q])=\begin{cases}\varepsilon, & \text{if } \#\delta(p, a)=1\\ c^{\#\delta(p, a)}d, & \text{otherwise.}\end{cases}$$

Then

$$L_i(A) = f(\{\pi \in R \mid \beta(\pi) \leqslant i\})$$
$$= f(R \cap g^{-1}(S_i)),$$

where

$$S_i = \{c^{j_1}d \cdots c^{j_t}d \mid t \geqslant 1, \text{ each } j_k \geqslant 2, j_1 j_2 \cdots j_t \leqslant i\} \cup \{\varepsilon\}$$

is a finite set. Hence, $L_i(A)$ is regular.

The preceding argument is effective; so from A and i, we can construct an automaton A_i for $L_i(A)$. Then $\beta_A = i$ iff $L(A_{i-1}) \neq L(A_i) = L(A)$, which is decidable since finite automata can be tested for equivalence. Thus, β_A is computable: if it is finite, just test it for equality with each positive integer in turn. ∎

3.2. THEOREM. *The spectrum of a regular language is computable.*

Proof. Any standard description of a regular language L can be converted to a deterministic finite automaton and then minimized to find $\sigma_1(L)$. There are (up to isomorphism) only finitely many finite automata having no more than $\sigma_1(L)$ states, and each can be tested to see if it accepts L. Hence, all pairs $(|A|, \beta_A)$, where $|A| \leqslant \sigma_1(L)$ and $L(A) = L$, can be found. If $(s_1, \beta_1), \ldots, (s_k, \beta_k)$, are the minimal pairs (where $(s, \beta) \leqslant (s', \beta')$ iff $s \leqslant s'$ and $\beta \leqslant \beta'$), with $1 = \beta_1 < \cdots < \beta_k$, then $s_1 > s_2 > \cdots > s_k$. If we set $\beta_{k+1} = \infty$, then the spectrum can be computed as follows: $\sigma_i = s_j$ for all i in the range $\beta_j \leqslant i < \beta_{j+1}$. ∎

4. UPPER BOUNDS ON SPECTRA

If a regular language L requires n states to be recognized by a nondeterministic finite automaton, i.e.,

$$\sigma(L) = (?, ?, ?, \ldots; n),$$

then we know from the subset construction that no more than $2^n - 1$ states are needed to recognize L with restricted amounts of nondeterminism. Hence, $2^n - 1$ serves as an upper bound for the other entries in the spectrum. In this section, we show that this upper bound is sharp, or at least nearly so. We begin with a lemma.

4.1. LEMMA. *Let L be a regular language and \$ a new symbol. Then*

$$\sigma((\$L\$)^*) = (k, k, k, \ldots; n)$$

for some k and n.

Proof. If $\sigma((\$L\$)^*) = (\sigma_1, \sigma_2, \ldots; \sigma_\infty)$, then $\sigma_1 \geqslant \sigma_2 \geqslant \cdots$, so it suffices to prove that $\sigma_i \leqslant \sigma_1$ for all finite i. For any such i, let $A = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton for $(\$L\$)^*$ with σ_i states and $\beta_A \leqslant i$. It suffices to construct a deterministic finite automaton B for $(\$L\$)^*$ with the same number of states.

Extend δ to sets in the usual way, $\delta(P, K) = \bigcup \{\delta(q, x) \mid q \in P, x \in K\}$; let

$$S = \delta(q_0, (\$L\$)^*),$$

$$T = \{q \in Q \mid \delta(q, (\$L\$)^*) \cap F \neq \varnothing\};$$

and let δ' be the deterministic part of δ,

$$\begin{aligned} \delta'(q, a) &= \delta(q, a), && \text{if } \#\delta(q, a) = 1 \\ &= \varnothing, && \text{otherwise.} \end{aligned}$$

CLAIM 1. $(\$L\$)^* = \{y \mid \delta'(S, y) \cap T \neq \varnothing\}$.

Proof of Claim 1. Suppose y is in $(\$L\$)^*$. Then y^i is also in $(\$L\$)^*$; but $\beta_A \leqslant i$, so some computation of A from q_0 to F has y^i as input and has fewer than i nondeterministic moves. But then some factor y in y^i is input to a computation of A from S to T having only deterministic moves. Hence, $\delta'(S, y) \cap T \neq \varnothing$.

Conversely, suppose y satisfies the condition $\delta'(S, y) \cap T \neq \varnothing$. Then $t \in \delta'(s, y) \subseteq \delta(s, y)$ for some $s \in S$, $t \in T$. By the definition of S and T, $s \in \delta(q_0, x)$ and $\delta(t, z) \cap F \neq \varnothing$ for some $x, z \in (\$L\$)^*$. Hence, $\delta(q_0, xyz) \cap F \neq \varnothing$, so $xyz \in (\$L\$)^*$. But x, z, xyz in $(\$L\$)^*$ imply y is in $(\$L\$)^*$.

This completes the proof of Claim 1.

Intuitively, Claim 1 means that $(\$L\$)^*$ can be recognized by an automaton with the same set of states as A but which is deterministic except in its choice of starting state. We now show that a single starting state suffices.

Let S_0 be a minimal subset of S such that Claim 1 remains true with S_0 in place of S.

CLAIM 2. $(\$L\$)^* = \{y \mid \delta'(s, y) \cap T \neq \varnothing\}$ *for all* $s \in S_0$.

Proof of Claim 2. Since $(\$L\$)^* = \{y \mid \delta'(S_0, y) \cap T \neq \varnothing\} \supseteq \{y \mid \delta'(s, y) \cap T \neq \varnothing\}$, $s \in S_0$, it suffices to show that, for all $s \in S_0$, $\delta'(s, y) \cap T \neq \varnothing$ if $y \in (\$L\$)^*$.

Suppose to the contrary that $\delta'(s_0, y_0) \cap T = \varnothing$ for some $s_0 \in S_0$, $y_0 \in (\$L\$)^*$. Then

$$\delta'(s_0, y_0(\$L\$)^*) \cap T = \varnothing. \qquad (*)$$

For if (*) were false then $q \in \delta'(s_0, y_0)$ and $\delta'(q, y) \cap T \neq \varnothing$ for some $q \in Q$

and $y \in (\$L\$)^*$. But $\delta'(q, y) \cap T \neq \varnothing$ for some $y \in (\$L\$)^*$ implies q is in T by the definition of T. So $q \in \delta'(s_0, y_0) \cap T = \varnothing$, a contradiction, proving (*).

Now, for all $y \in (\$L\$)^*$, $y_0 y$ is in $(\$L\$)^*$. Hence, since Claim 1 remains true for S_0, $\delta'(S_0, y_0 y) \cap T \neq \varnothing$. But then it follows from (*) that $\delta'(S_0 - \{s_0\}, y_0 y) \cap T \neq \varnothing$. Thus, $\delta'(S_0', y) \cap T \neq \varnothing$, where $S_0' = \delta'(S_0 - \{s_0\}, y_0) \subseteq \delta'(S, (\$L\$)^*) \subseteq S$. So

$$(\$L\$)^* \subseteq \{y \mid \delta'(S_0', y) \cap T \neq \varnothing\} \subseteq \{y \mid \delta'(S, y) \cap T \neq \varnothing\} = (\$L\$)^*.$$

But then Claim 1 is also true of S_0', although $\# S_0' \leqslant \#(S_0 - \{s_0\}) < \# S_0$. This contradicts the minimality of S_0 and proves Claim 2.

Finally, let $B = (Q, \Sigma, \delta', s_0, T)$ for any $s_0 \in S_0$. Then B accepts $(\$L\$)^*$ by Claim 2, and B is deterministic as required. ∎

We can now establish that the upper bound on spectra discussed earlier is nearly sharp.

4.2. THEOREM. *For every regular language L,*

$$\sigma(L) \leqslant (2^n - 1, 2^n - 1, \ldots, 2^n - 1, \ldots; n),$$

where[2] $n = \sigma_\infty(L)$. *Furthermore, this bound is approximately the best possible in the sense that, for each $n \geqslant 1$, there is a regular language L_n with*

$$\sigma(L_n) = (2^{n-1}, 2^{n-1}, \ldots, 2^{n-1}, \ldots; n).$$

Proof. Since we showed earlier that $\sigma_i \leqslant 2^{\sigma_\infty} - 1$ for $i < \infty$ in any spectrum, it suffices to exhibit the L_n. If $n = 1$ then $L_1 = \varnothing$ has the required property, so assume $n > 1$. Let R_n be a nonempty regular language that can be defined by an $(n-1)$-state nondeterministic automaton but whose minimal complete deterministic automaton has 2^{n-1} states (Meyer and Fischer, 1971). Thus, $\sigma_\infty(R_n) = n - 1$ and $\sigma_1(R_n) = 2^{n-1} - 1$. Then let $L_n = (\$R_n\$)^*$, \$ a new symbol.

We now compute the spectrum of L_n. If $A = (Q, \Sigma, \delta, q_0, F)$ is a finite automaton for R_n then we can construct a finite automaton A' for $L_n = (\$R_n\$)^*$ having one more state p_0 by defining $A' = (Q \cup \{p_0\}, \Sigma \cup \{\$\}, \delta', p_0, \{p_0\})$, where

$$\delta'(p_0, \$) = \{q_0\}$$

$$\delta'(p_0, a) = \varnothing, a \in \Sigma,$$

$$\delta'(q, a) = \delta(q, a), q \in Q, a \in \Sigma$$

$$\delta'(q, \$) = \{p_0 \mid q \in F\}, q \in Q.$$

[2] Sequences are compared elementwise, so this means that $\sigma_i(L) \leqslant 2^n - 1$, $1 \leqslant i < \infty$, and $\sigma_\infty(L) \leqslant n$.

Since A' is deterministic if A is,

$$\sigma_{\infty}(L_n) \leqslant \sigma_{\infty}(R_n) + 1 = n$$

and

$$\sigma_1(L_n) \leqslant \sigma_1(R_n) + 1 = 2^{n-1}.$$

But it is well known that the minimal number of states in a complete (resp. incomplete) deterministic automaton for a regular language R over an alphabet Σ is the number of (nonempty) left quotients $x \backslash R$ of R, $x \in \Sigma^*$. Since, for any string x not containing \$, $(\$x)\backslash(\$R_n\$)^* = (x \backslash R_n)\$(\$R_n\$)^*$, it follows that $L_n = (\$R_n\$)^*$ has at least as many nonempty left quotients as R_n, and since $L_n = \varepsilon \backslash L_n$ is still another left quotient, it has at least one more. Hence,

$$\sigma_1(L_n) \geqslant \sigma_1(R_n) + 1 = 2^{n-1}.$$

Therefore, $\sigma_1(L_n) = 2^{n-1}$. But then $2^{n-1} = \sigma_1(L_n) \leqslant 2^{\sigma_\infty(L_n)} - 1$, so $n - 1 < \sigma_\infty(L_n)$. Therefore, $\sigma_\infty(L_n) = n$. Hence, by Lemma 4.1,

$$\sigma(L_n) = (2^{n-1}, 2^{n-1}, ..., 2^{n-1}, ...; n). \quad \blacksquare$$

5. Lower Bounds on Spectra

The previous section began with the observation that, if a regular language L can be recognized by a nondeterministic automaton with n states then it can be recognized by an incomplete deterministic automaton requiring at most $2^n - 1$ states. This implies that if $\sigma_\infty \leqslant n$ then $\sigma_1 \leqslant 2^n - 1$ (and so every $\sigma_i \leqslant 2^n - 1$, which yields an upper bound on the spectrum). On the other hand, if L can be recognized nondeterministically with fewer than n states then it can be recognized deterministically with at most $2^{n-1} - 1$ states. So if $\sigma_1 \geqslant 2^n - 1$, then $\sigma_1 > 2^{n-1} - 1$, so $\sigma_\infty \geqslant n$, and hence every $\sigma_i \geqslant n$. This lower bound, however, is not sharp. To obtain a nearly sharp lower bound, we improve on the subset construction when just a small amount of nondeterminism is present. The intuitive idea is that, if an automaton A for L has branching $\beta \leqslant k$, then A can be simulated deterministically by k automata acting in parallel, each using the same starting state as A, in the following way: when A is in state p and encounters an r-way branch, those of the simulating machines that are in state p split into r equal groups and each group proceeds along one of the branches. Since $\beta \leqslant k$, k automata are enough to simulate all the computations of A that need to be simulated. This is formalized in Lemma 5.2. First, we prove a technical result to the effect that, in a series of integer divisions, truncation may be postponed until the end.

5.1. LEMMA. *For any integers* $k \geqslant 0$, $m \geqslant 1$, $n \geqslant 1$, $\lfloor \lfloor k/m \rfloor / n \rfloor = \lfloor k/mn \rfloor$.

Proof. Let $k = mq_1 + r_1$, $0 \leqslant r_1 < m$, and $q_1 = nq_2 + r_2$, $0 \leqslant r_2 < n$. Then $\lfloor \lfloor k/m \rfloor / n \rfloor = \lfloor q_1/n \rfloor = q_2$. But $k = mnq_2 + mr_2 + r_1$, where $mr_2 + r_1 < m(n-1) + m = mn$, so $\lfloor k/mn \rfloor = q_2$. ∎

5.2. LEMMA. *For each finite automaton* A *with* $\beta_A \leqslant k$, *there is a deterministic finite automaton accepting the same language, whose state set is the family of multisets of states of* A (*i.e., sets with multiple entries*) *of size* k.

Proof. If f is a multiset, let $f(p)$ be the number of times that p occurs in f. Thus, $\#f = \sum_p f(p)$. Let $A = (Q, \Sigma, \delta, q_0, F)$. We first construct a complete deterministic finite automaton $B = (Q', \Sigma, \delta', q_0', F')$ accepting the same language as A, where Q' is the family of multisets of states in Q of size less than or equal to k. Define

$$q_0'(p) = k, \qquad \text{if} \quad p = q_0$$
$$= 0, \qquad \text{otherwise;}$$

$$F' = \{ f \in Q' \mid f(p) \neq 0 \text{ for some } p \in F \};$$

$$(\delta'(f, a))(q) = \sum \left\lfloor \frac{f(p)}{\#\delta(p, a)} \right\rfloor,$$

where the summation is taken over all p such that $q \in \delta(p, a)$, and where, of course, the empty summation is zero. If f is in Q' and $g = \delta'(f, a)$ then

$$\#g = \sum_q (\delta'(f, a))(q) \leqslant \sum_p \sum_{q \in \delta(p,a)} \frac{f(p)}{\#\delta(p, a)} = \sum_p f(p) = \#f \leqslant k,$$

so g is also in Q'. Thus, δ' is well defined. To show that B accepts the same language as A, we first establish the following claim.

CLAIM. *If there is a computation in* A *from* p *to* q *with input* x *and branching* k', *then for each* $f \in Q'$, $(\delta'(f, x))(q) \geqslant \lfloor f(p)/k' \rfloor$.

Proof of Claim. Let π be the computation in A, so that π is a string of moves, and proceed by induction on the length $|\pi|$ of π.

If $|\pi| = 0$ then $x = \varepsilon$, $p = q$, $k' = 1$, $\delta'(f, x) = f$, and the claim is trivially true.

If $|\pi| = 1$ then $x = a$, $a \in \Sigma$, and $k' = \#\delta(p, a)$, and the claim follows from the definition of δ'.

If $|\pi| > 1$ then $\pi = \pi_1 \pi_2$, where π_1 and π_2 are shorter than π. If x_i and k_i are the input and branching of π_i then $x = x_1 x_2$ and $k' = k_1 k_2$. If r is the

transition state between π_1 and π_2 then by the inductive hypothesis, for $f \in Q'$,

$$(\delta'(f, x_1))(r) \geqslant \lfloor f(p)/k_1 \rfloor$$

and

$$(\delta'(\delta'(f, x_1), x_2))(q) \geqslant \lfloor (\delta'(f, x_1))(r)/k_2 \rfloor.$$

Hence,

$$\begin{aligned}(\delta'(f, x))(q) &= (\delta'(\delta'(f, x_1), x_2))(q) \\ &\geqslant \lfloor \lfloor f(p)/k_1 \rfloor / k_2 \rfloor \\ &= \lfloor f(p)/k' \rfloor, \qquad \text{by Lemma 5.1.}\end{aligned}$$

This completes the proof of the claim.

Next, to prove that $L(B) = L(A)$, i.e., that B and A accept the same language, suppose that x is in $L(A)$. Then there is a computation in A from q_0 to some $q \in F$ with input x and degree of branching $k' \leqslant k$. By the claim,

$$(\delta'(q_0', x))(q) \geqslant \left\lfloor \frac{q_0'(q_0)}{k'} \right\rfloor = \left\lfloor \frac{k}{k'} \right\rfloor \neq 0,$$

so $\delta(q_0', x)$ is in F'. Hence, x is in $L(B)$ and so $L(A) \subseteq L(B)$. On the other hand, it follows from the definition of δ' that, if $(\delta'(f, a))(q) \neq 0$, then q is in $\delta(p, a)$ for some p with $f(p) \neq 0$. This easily extends from single letters a to strings x, and hence $L(B) \subseteq L(A)$. So $L(B) = L(A)$.

Finally, let Q'' be the family of multisets of states of A of size exactly k. To obtain a deterministic automaton for $L(A)$ with state set Q'', we proceed as follows. For any nonempty multiset f which is in Q' (so that $\# f \leqslant k$), let f_* in Q'' be any multiset of size exactly k such that, for each p, either $f_*(p) \geqslant f(p) > 0$ or $f_*(p) = f(p) = 0$. For $f \in Q''$, define $\delta''(f, a)$ as

$$\delta''(f, a) = \delta'(f, a)_* \qquad \text{if} \quad \delta'(f, a) \neq \varnothing,$$

and $\delta''(f, a)$ is undefined otherwise. Then $C = (Q'', \Sigma, \delta'', q_0', F' \cap Q'')$ is a deterministic finite automaton and it may be verified that $L(C) = L(B) = L(A)$, as required. ∎

5.3. COROLLARY. *For any nontrivial spectrum,*

$$\sigma_1 \leqslant \binom{\sigma_i + i - 1}{i} < \sigma_i^i, \qquad 1 < i < \infty.$$

Proof. Suppose A is an automaton having σ_i states and $\beta_A \leqslant i$. Then by Lemma 5.2, there is a deterministic automaton B for the same language whose states are the multisets of states of A of size i. But the multisets of size i with elements chosen from a set of size σ_i are in one-to-one correspondence with sequences of σ_i nonnegative integers, written in unary notation, separated by commas, and summing to i; and the number of these is the number of strings of ones and commas containing exactly i ones and $\sigma_i - 1$ commas, which is

$$\frac{(\sigma_i + i - 1)!}{i!(\sigma_i - 1)!} = \binom{\sigma_i + i - 1}{i}.$$

Hence,

$$\sigma_1 \leqslant |B| = \binom{\sigma_i + i - 1}{i},$$

establishing the first inequality. The second inequality follows from the fact that $\sigma_i > 1$ if the spectrum is nontrivial, so for $i > 1$,

$$\binom{\sigma_i + i - 1}{i} = \left(\frac{\sigma_i + i - 1}{i}\right)\left(\frac{\sigma_i + i - 2}{i-1}\right) \cdots \left(\frac{\sigma_i}{1}\right) < \sigma_i^i. \quad \blacksquare$$

We can now establish a nearly sharp lower bound on spectra.

5.4. THEOREM. *For any regular language L, if $\sigma_1(L) \geqslant 2^n - 1 > 1$ then*

$$\sigma_i(L) \geqslant 2^{n/i} \quad \text{for} \quad 1 < i \leqslant n/\log_2 n,$$

$$\sigma_i(L) \geqslant n \quad \text{for} \quad n/\log_2 n \leqslant i \leqslant \infty,$$

and these lower bounds are approximately the best possible in the sense that, for each $n > 1$, there is a regular language L_n such that

$$\sigma_1(L_n) = 2^n,$$

$$\sigma_i(L_n) < 2i \cdot 2^{n/i} \quad \text{for} \quad 1 < i \leqslant n,$$

$$\sigma_i(L_n) < 4n \quad \text{for} \quad n \leqslant i < \infty,$$

$$\sigma_\infty(L_n) = n + 1.$$

Remark. Using less precise notation, we can say that

$$(2^n - 1, 2^{n/2}, 2^{n/3}, \ldots, n, \ldots; n) \leqslant \sigma(L)$$

serves as a lower bound whenever $2^n - 1 \leqslant \sigma_1(L)$, and for $n > 1$, $\sigma(L_n)$ approximates this bound, since

$$(2^n - 1, 2^{n/2}, 2^{n/3}, \ldots, n, n, \ldots; n) \leqslant \sigma(L_n)$$

$$\leqslant (2^n, 4 \cdot 2^{n/2}, 6 \cdot 2^{n/3}, \ldots, 4n, 4n, \ldots; n + 1).$$

Thus, for example, if $n = 5$, the lower bound is

$$(31, 5.6568..., 5, 5, ...; 5)$$

and, since the entries in a spectrum are integers,

$$(31, 6, 5, 5, ...; 5)$$

will in fact serve as a lower bound in this case. In effect, the theorem says that the most rapid rate at which a spectrum can decrease is for the ith entry ($i = 1, 2, ...$) to be essentially just the ith root of the first until the maximum (exponential) decrease has been attained after about $n/\log_2 n$ steps.

Proof. Any spectrum with $\sigma_1 \geqslant 2^n - 1 > 1$ is nontrivial, so by Corollary 5.3, $2^n - 1 \leqslant \sigma_1 < \sigma_i^i$, so $\sigma_i^i \geqslant 2^n$ and $\sigma_i \geqslant 2^{n/i}$ for $1 < i < \infty$. And $\sigma_i \geqslant \sigma_\infty \geqslant n$ since $\sigma_1 \geqslant 2^n - 1$. This establishes the lower bound in the theorem.

Next, we define the languages L_n and estimate their spectra. For any $n \geqslant 1$, let $L_n = \Sigma^* 1 \Sigma^{n-1}$, where $\Sigma = \{0, 1\}$, and consider its spectrum. Since L_n can obviously be recognized by a nondeterministic automaton with $n+1$ states, $\sigma_\infty \leqslant n+1$. Furthermore, it is easy to see that each string in Σ^n must drive a deterministic automaton for L_n to a different state, so $\sigma_1 \geqslant 2^n$. Hence, $2^n \leqslant \sigma_1 \leqslant 2^{\sigma_\infty} - 1$, and so $\sigma_\infty = n+1$. While it is not hard to prove that $\sigma_1 = 2^n$ by constructing a deterministic finite automaton that remembers its last n inputs, we are also interested in the middle entries σ_i of the spectrum. Thus, we want more generally to construct an automaton to recognize L_n with the benefit of one i-way branch using approximately $i2^{n/i}$ states. For $i > 1$, this generalizes the deterministic construction. The idea is to guess the length of the input modulo i and to remember just a suitable fraction $1/i$ of the last n inputs.

For fixed $i \geqslant 1$, let $m = \lceil n/i \rceil$ and let $\langle j \rangle$ be the remainder when any integer j is divided by i. We define a ("generalized") automaton $A = (Q, \Sigma, \delta, S, F)$ for L_n that has a deterministic transition function δ but an entire set S of start states:

$$Q = \{[x, \langle j \rangle] \mid x \in \Sigma^m, j \text{ an integer}\};$$

$$S = \{[0^m, \langle j \rangle] \mid j \text{ an integer}\};$$

$$F = \{[1x, \langle n \rangle] \mid x \in \Sigma^{m-1}\};$$

$$\delta([bx, \langle 0 \rangle], a) = [xa, \langle 1 \rangle] \qquad \text{for} \quad a, b \in \Sigma;$$

$$\delta([x, \langle j \rangle], a) = [x, \langle j+1 \rangle] \qquad \text{for} \quad a \in \Sigma, \langle j \rangle \neq \langle 0 \rangle.$$

To show that A accepts L_n, consider a string z of the form

$$z = xa_1 y_1 a_2 y_2 \cdots a_m y_m,$$

$$x \in \Sigma^*, \quad a_j \in \Sigma, \quad |a_1 y_1| = \cdots = |a_{m-1} y_{m-1}| = i, \qquad 0 < |a_m y_m| \leqslant i.$$

It is easily seen from the definition of δ that

$$\delta([0^m, \langle -|x| \rangle], z) = [a_1 a_2 \cdots a_m, \langle |z| - |x| \rangle],$$

which is in F iff $a_1 = 1$ and $\langle |z| - |x| \rangle = \langle n \rangle$. But if $\langle |z| - |x| \rangle = \langle n \rangle$ then $\langle p \rangle = \langle 0 \rangle$, where

$$\begin{aligned} p &= |z| - |x| - n \\ &= |a_1 y_1 \cdots a_m y_m| - n \\ &= (m-1)i + |a_m y_m| - n \\ &= (\lceil n/i \rceil i - n) - (i - |a_m y_m|). \end{aligned}$$

But $0 \leqslant \lceil n/i \rceil - n/i < 1$, so $0 \leqslant \lceil n/i \rceil i - n < i$; and $0 < |a_m y_m| \leqslant i$, so $-i < -(i - |a_m y_m|) \leqslant 0$. Hence, $-i < p < i$. Since $\langle p \rangle = 0$, it follows that $p = 0$. Thus, $|a_1 y_1 \cdots a_m y_m| = n$. So

$$\delta([0^m, \langle -|x| \rangle], z) \in F \qquad \text{iff} \quad z \in L_n.$$

From this, it follows that A accepts L_n.

Now A has $2^m i$ states and i start states. For $i = 1$, A is a deterministic finite automaton with 2^n states, so $\sigma_1 = 2^n$. For $i > 1$, A can be modified in the obvious way to have single start state by introducing one new state, and then $\beta_A = i$, since the start state makes an i-way branch and never recurs. We now estimate the number of states in A. It is easily verified that $2^x \leqslant 1 + x$ for all real x, $0 \leqslant x \leqslant 1$; hence,

$$2^{(i-1)/i} \leqslant 1 + \frac{i-1}{i},$$

and so

$$2^{(i-1)/i} i + 1 \leqslant 2i.$$

Therefore,

$$\begin{aligned} 2^m i + 1 &= 2^{\lceil n/i \rceil} i + 1 \\ &\leqslant 2^{n/i + (i-1)/i} i + 1 \\ &< 2^{n/i}(2^{(i-1)/i} i + 1) \\ &\leqslant 2^{n/i} \cdot 2i, \qquad \text{by the inequality above.} \end{aligned}$$

Hence, $\sigma_i < 2i \cdot 2^{n/i}$ for $1 < i < \infty$. And for $n \leqslant i < \infty$, $\sigma_i \leqslant \sigma_n < 2n \cdot 2^{n/n} = 4n$. ∎

6. Conclusions

Although a good understanding of nondeterminism in finite automata should be much easier to achieve than in Turing machines, the subject has received relatively little study and many questions remain unanswered. In the present study, we have restricted attention to recognizing regular languages using finite automata that employ just a bounded amount of nondeterminism, and we investigated the effect that changing this bound has on the sizes of the automata. This led to the concept of the spectrum of a regular languge, and we obtained nearly sharp upper and lower bounds on these spectra. Among the problems remaining unresolved about spectra, we list these:

1. Can more precise necessary and sufficient conditions be found for a sequence to be spectrum of a regular language than merely that it be monotone nonincreasing and lie between the upper and lower bounds of Sections 4 and 5? Can necessary and sufficient conditions be found?

2. Are other properties of regular languages reflected in their spectra?

3. In this paper, no use was made of finite automata with behavior intermediate between bounded nondeterminism and unlimited nondeterminism, such as automata allowed to have $O(n^{1/2})$ guesses on inputs of length n (see Goldstine, Leung and Wotschke, 1989, and Simon, 1987). Are there regular languages whose behavior with respect to nondeterminism can only be captured adequately by considering such automata?

Received July 12, 1988; final manuscript received May 25, 1989

References

Goldstine, J., Leung, H., and Wotschke, D. (1989), "On the Relation between Ambiguity and Nondeterminism in Finite Automata," CS-89-19, The Pennsylvania State University, University Park, Pennsylvania.

Hashiguchi, K. (1982), Limitedness theorem on finite automata with distance functions, *J. Comput. System Sci.* **24**, No. 2, 233–244.

Kintala, C. M. R., and Fischer, P. C. (1980), Refining nondeterminism in relativized polynomial-time bounded computations, *SIAM J. Comput.* **9**, No. 1, 46–53.

Kintala, C. M. R., and Wotschke, D. (1980), Amount of nondeterminism in finite automata, *Acta Inform.* **13**, 199–204.

Leung, H. (1988), On the topological structure of a finitely generated semigroup of matrices, *Semigroup Forum* **37**, No. 3, 273–287.

Meyer, A. R., and Fischer, M. J. (1971), Economies of description by automata, grammars, and formal systems, *in* "Proceedings, 12th SWAT Symposium," pp. 188–191.

Savitch, W. J., and Vermeir, D. (1981), On the amount of nondeterminism in pushdown automata, *Fund. Inform.* **4**, No. 4, 401–418.

Simon, I. (1987), "The Nondeterministic Complexity of a Finite Automaton," RT-MAP-8703, University of Sao Paulo, Sao Paulo, Brazil.